



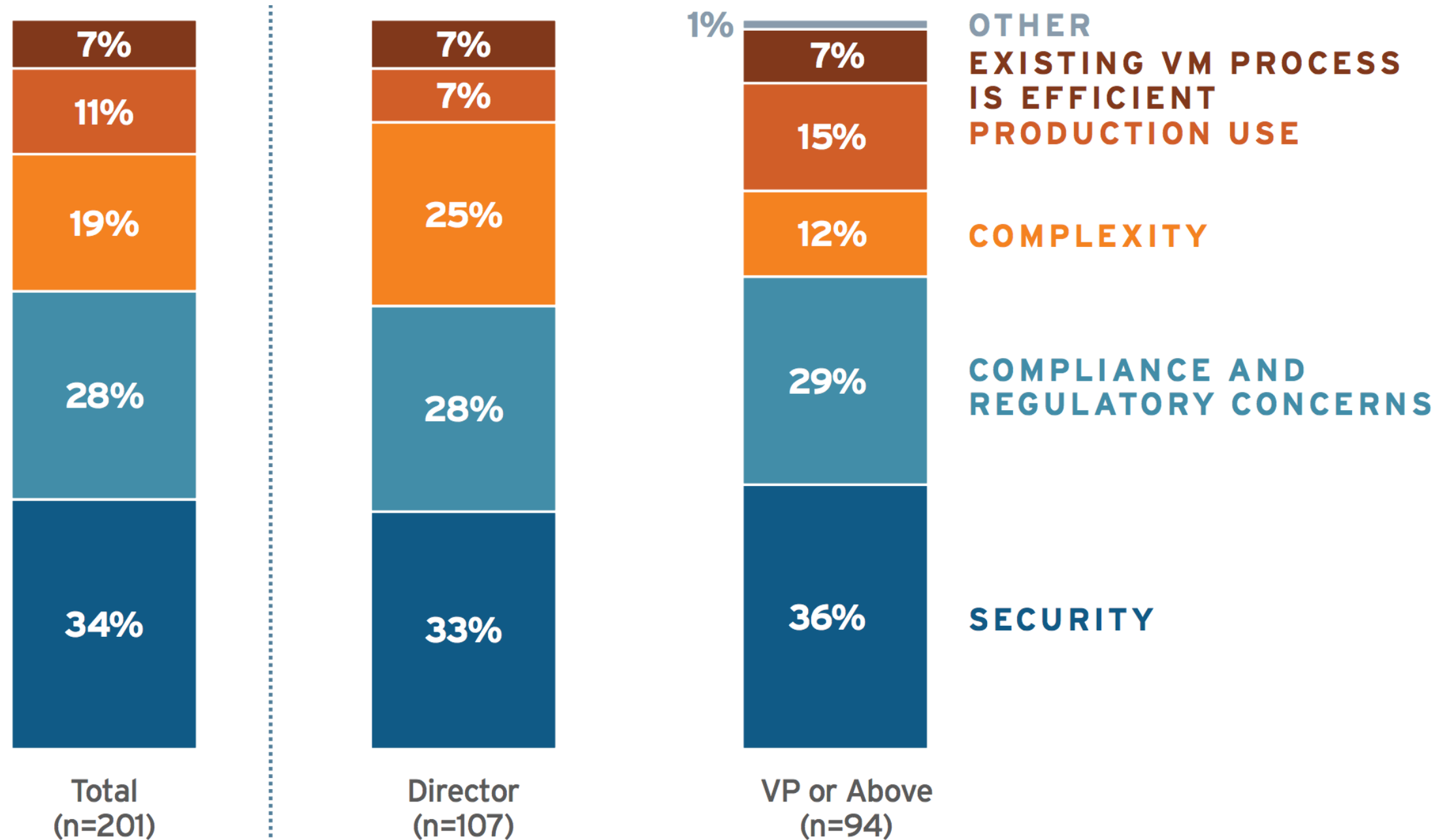
Containers for the underprivileged

With APP SWITCH

Dinesh Subhraveti

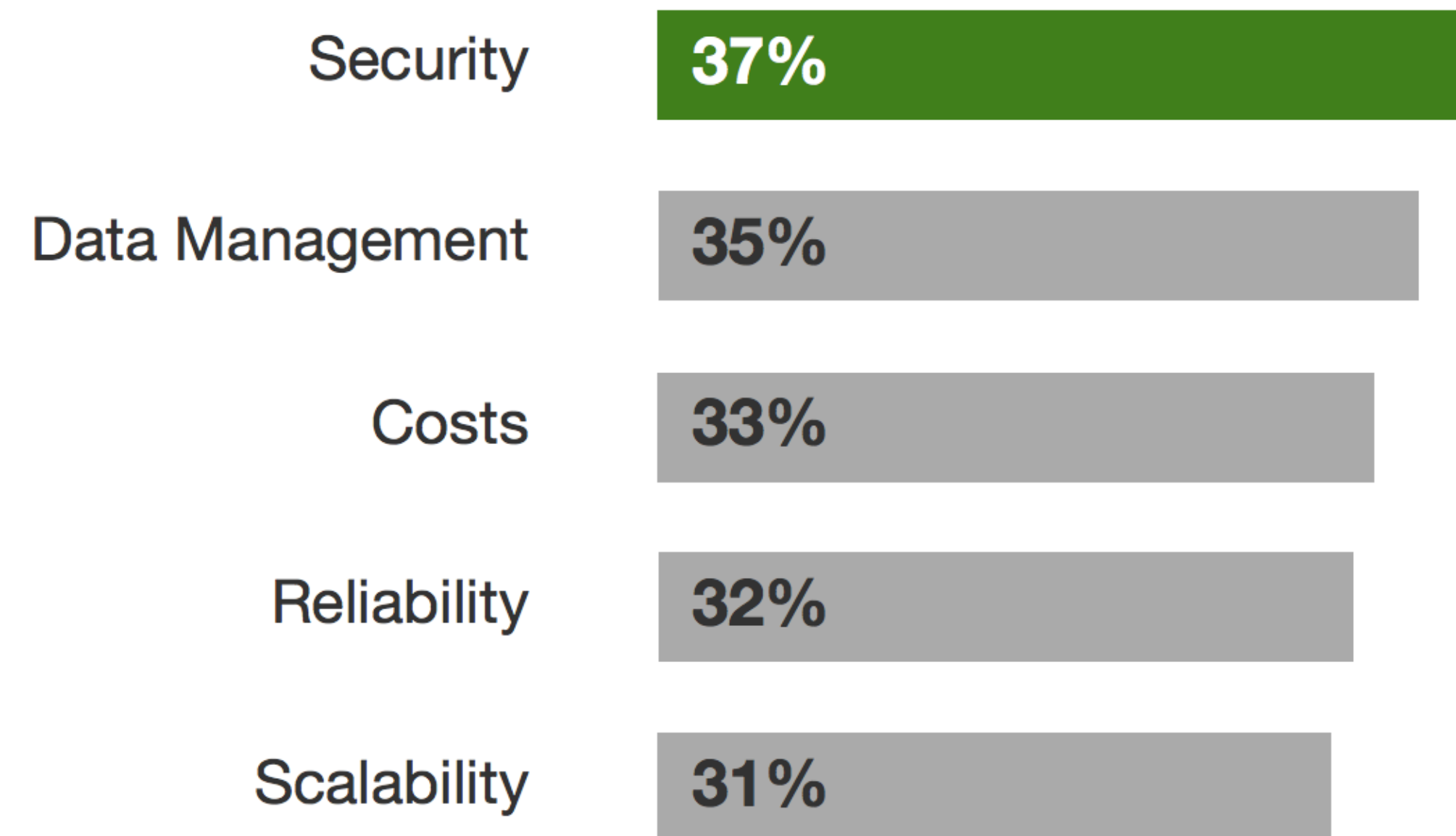
@dsubhraveti

Security — key barrier for container adoption



“What is the biggest hurdle to container adoption in your organization”

[Source: 451 Research](#)



“What were the biggest roadblocks/challenges that you experienced when deploying container technology?”

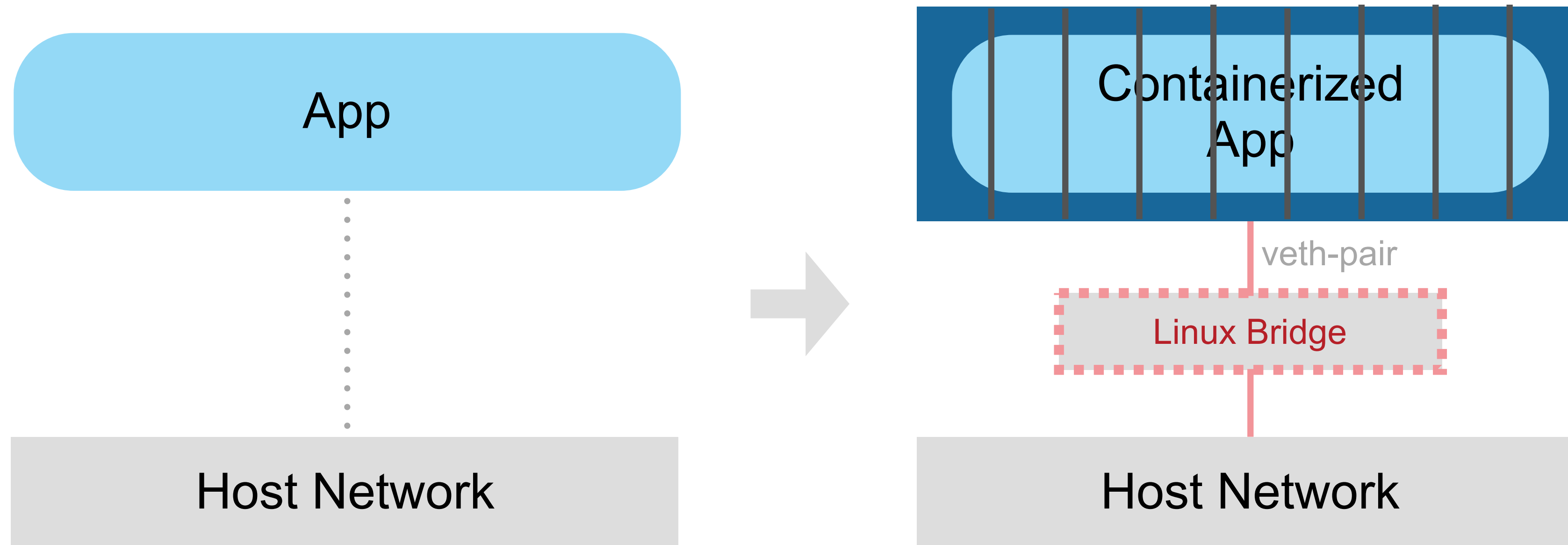
[Source: Forrester Consulting](#)

Container? No such thing!

- ▶ No container object in the kernel – It's just a transparent layer
 - ▶ Then how is it that running an app as container is so different from running it "natively"
- ▶ Let's define a couple desired properties for the container abstraction
 - ▶ **Conservation of privilege:** *I shouldn't need additional privilege to run my app as container*
 - ▶ **Conservation of resources:** *Resources I already possess shouldn't be taken away*

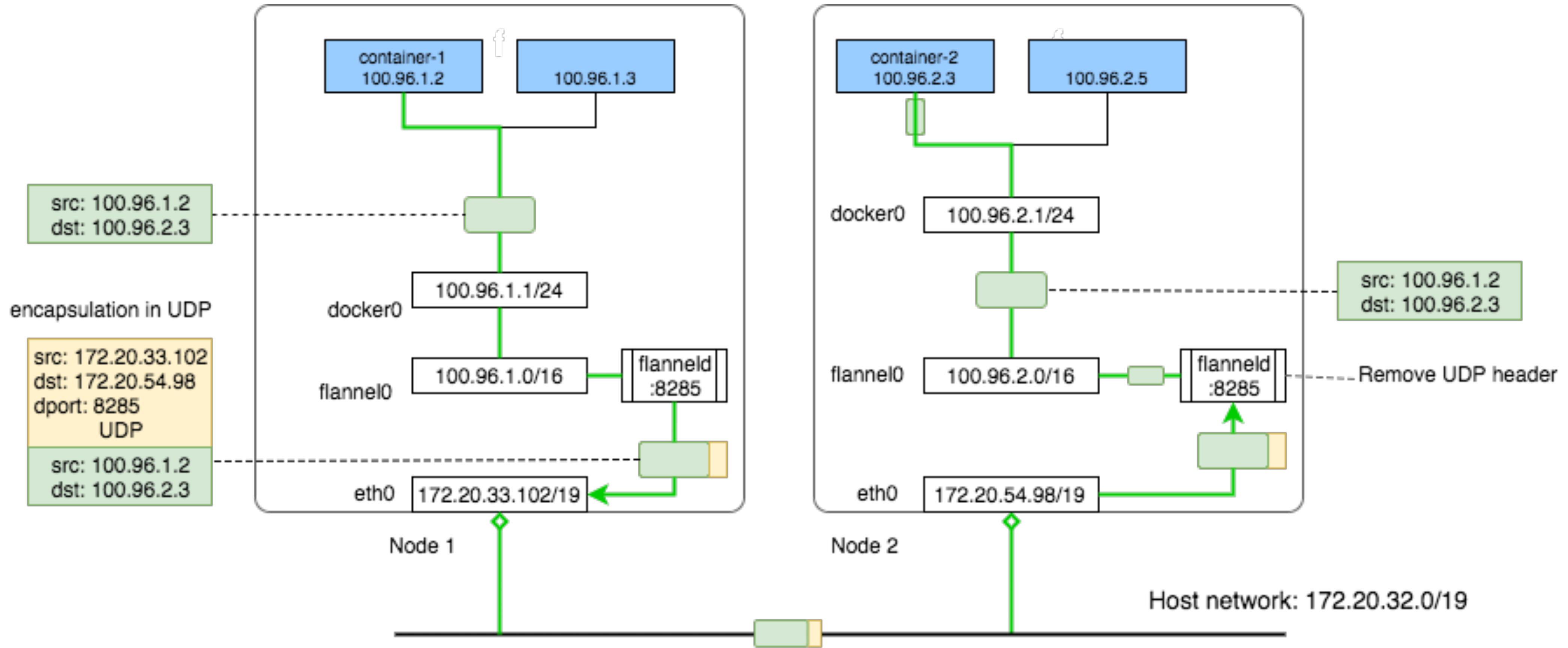
Current design of containers violates both

Privilege Drift

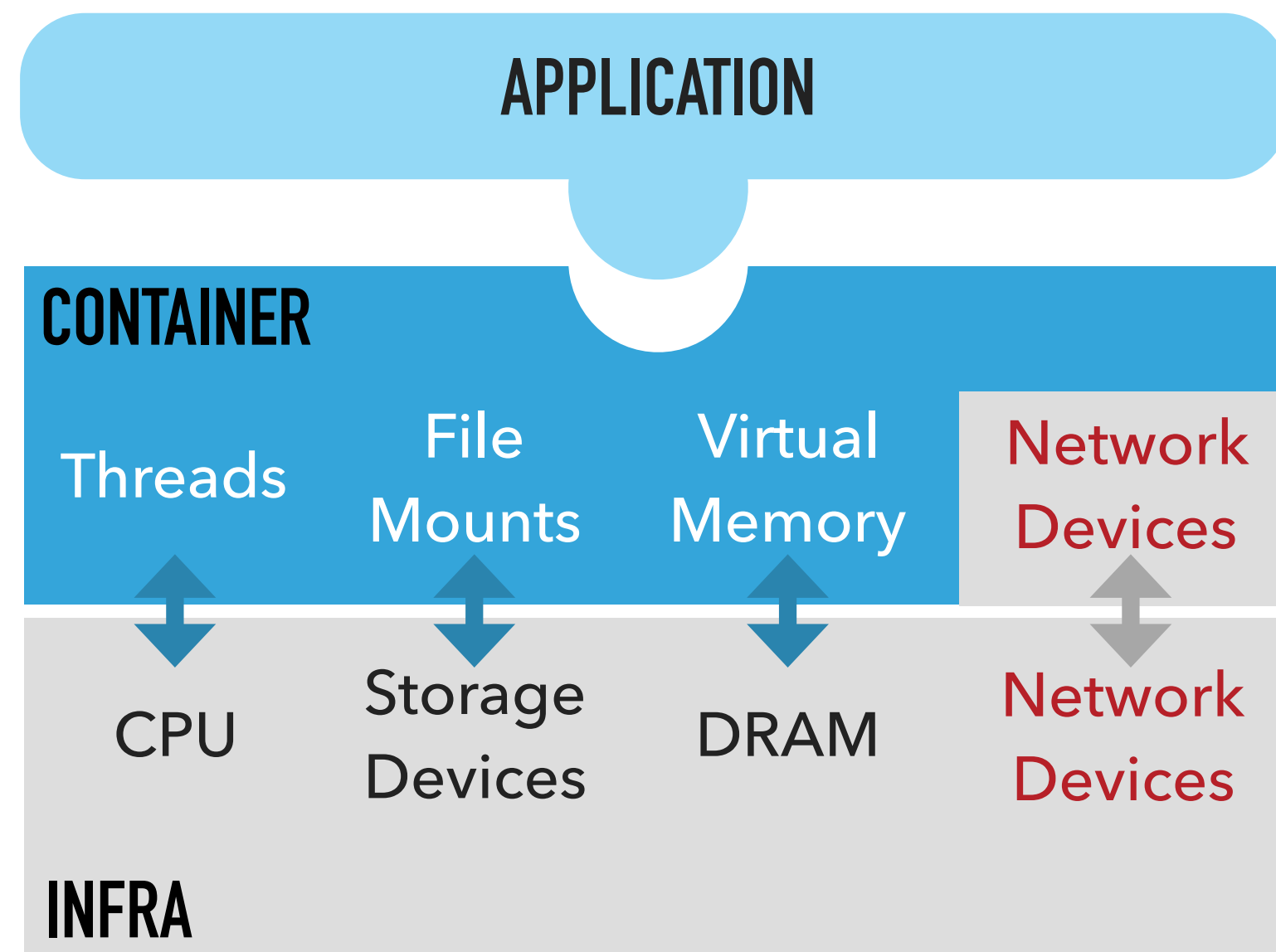


Low-level mechanisms that plumb host network into container require privilege
Ideally, privilege footprint of the container should be same as that of app

Container Networking is Broken

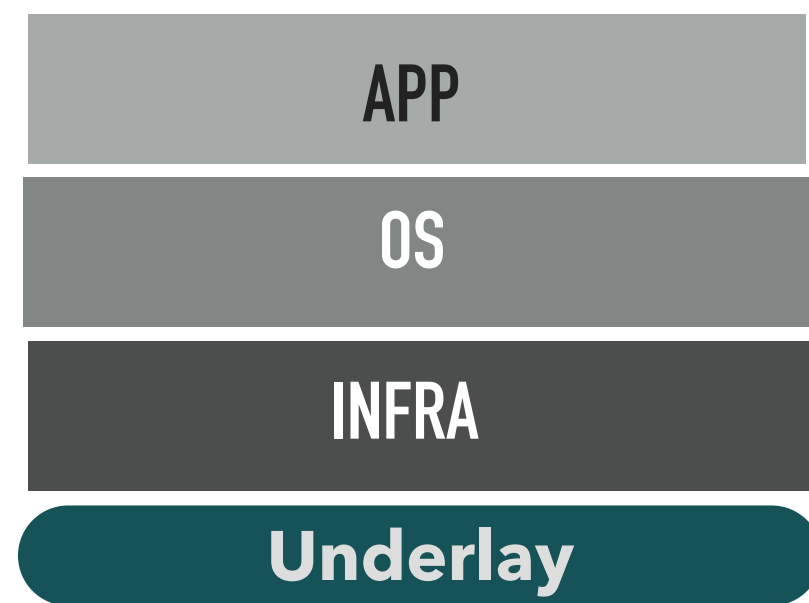


Container Networking is Broken

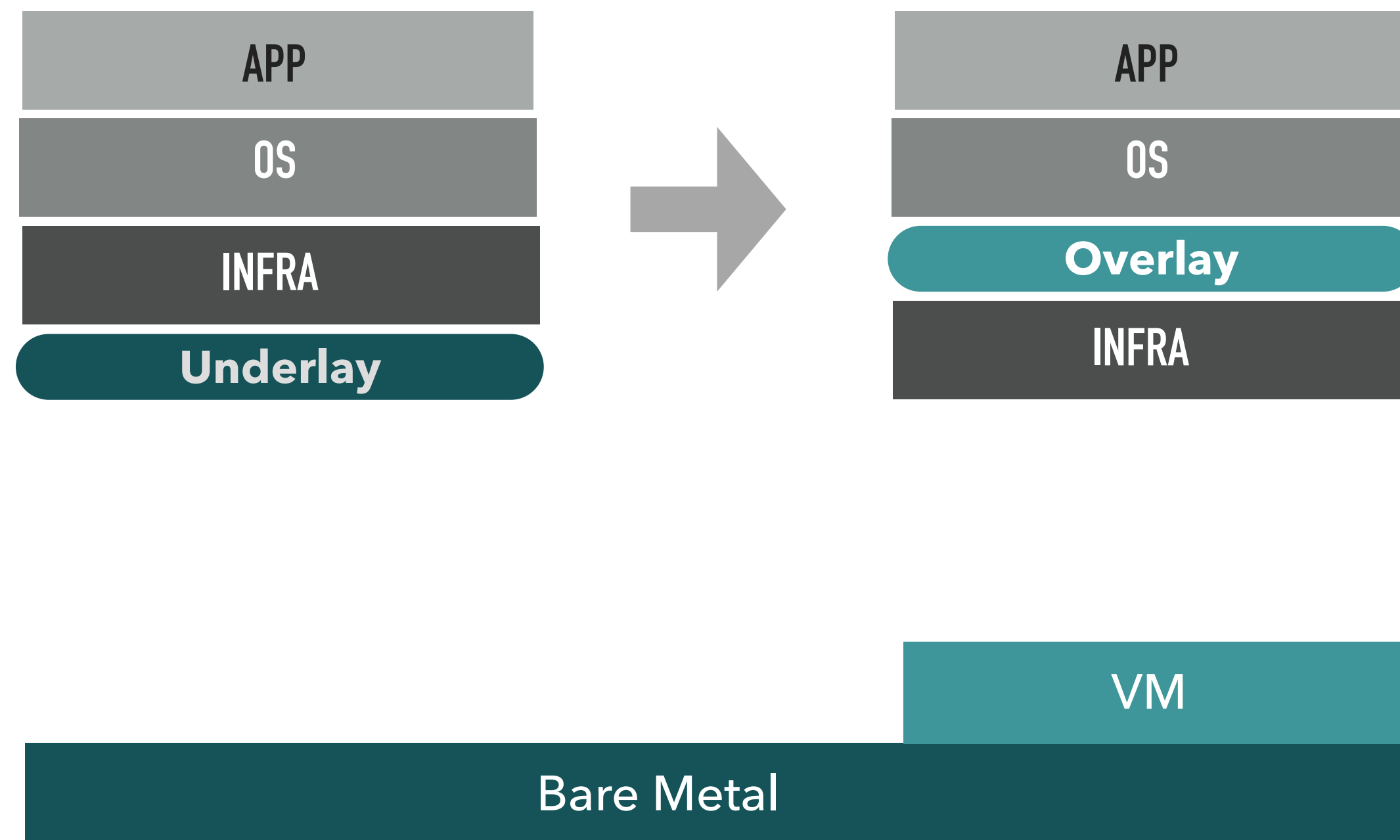


Container uses OS level abstractions for all resources except network and ends up requiring privilege

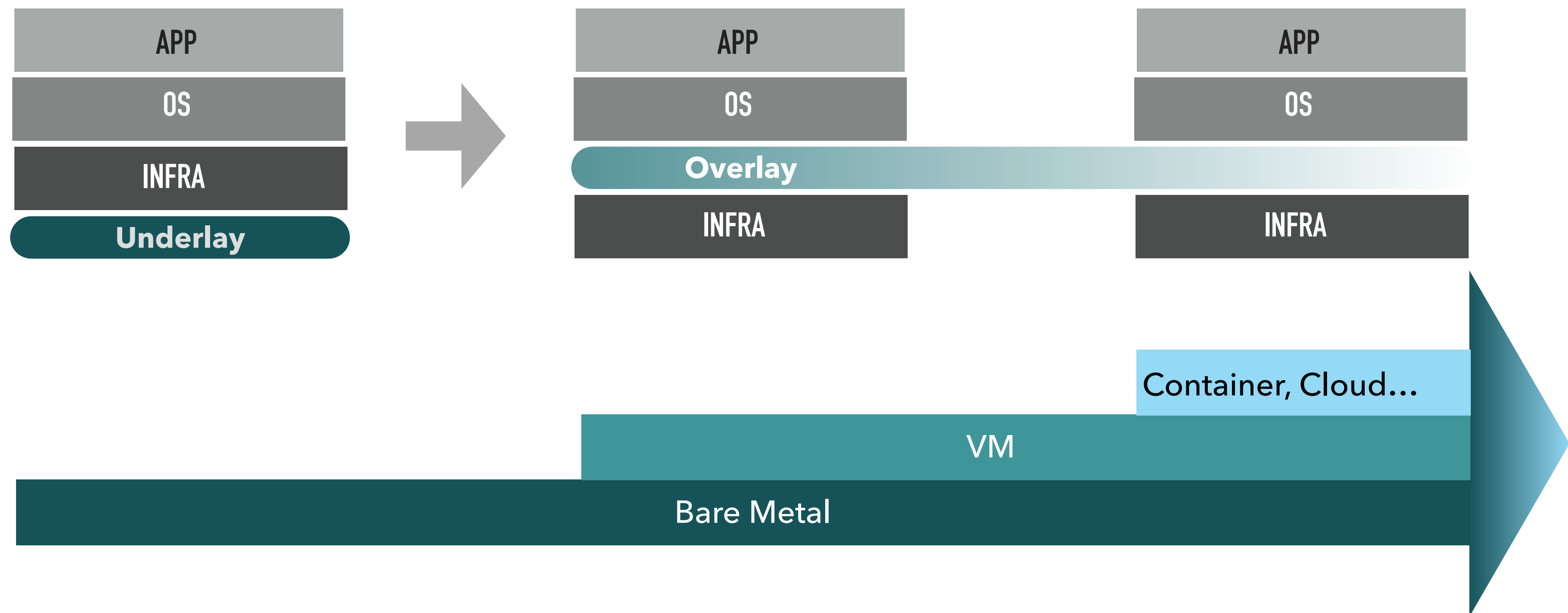
Evolution of network abstraction



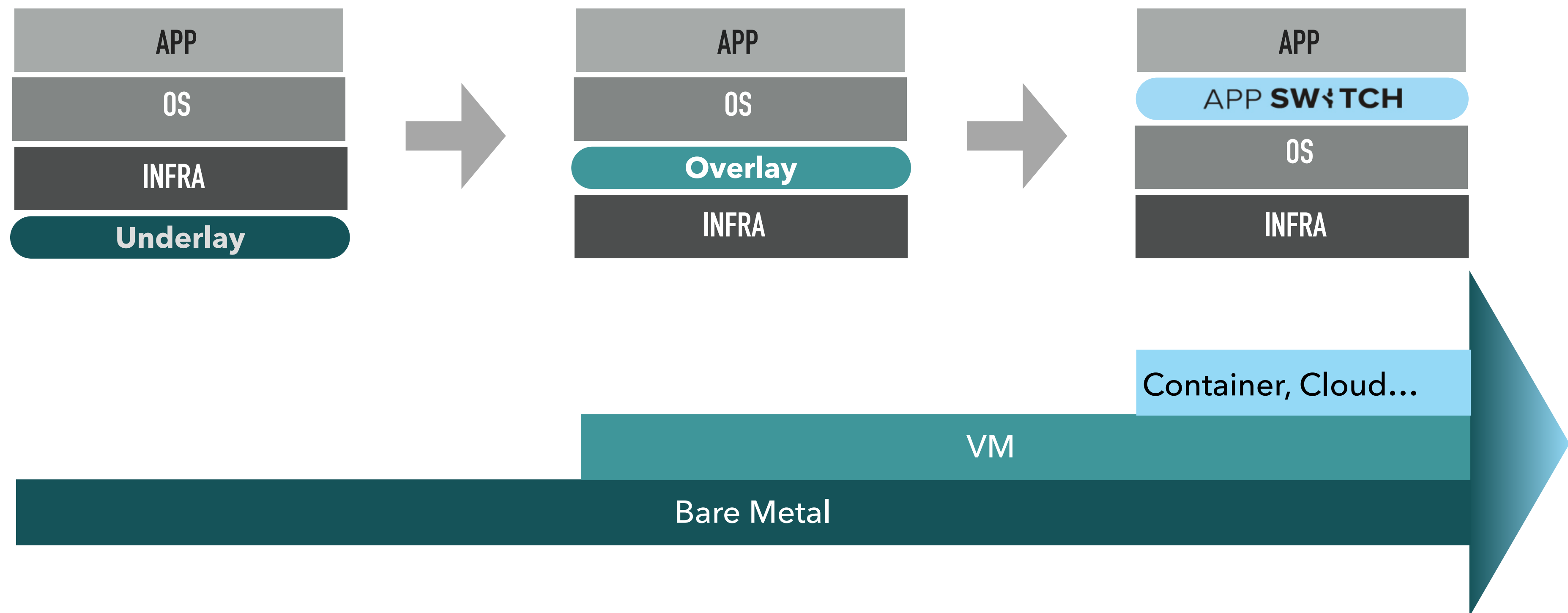
Evolution of network abstraction



Evolution of network abstraction



Evolution of network abstraction

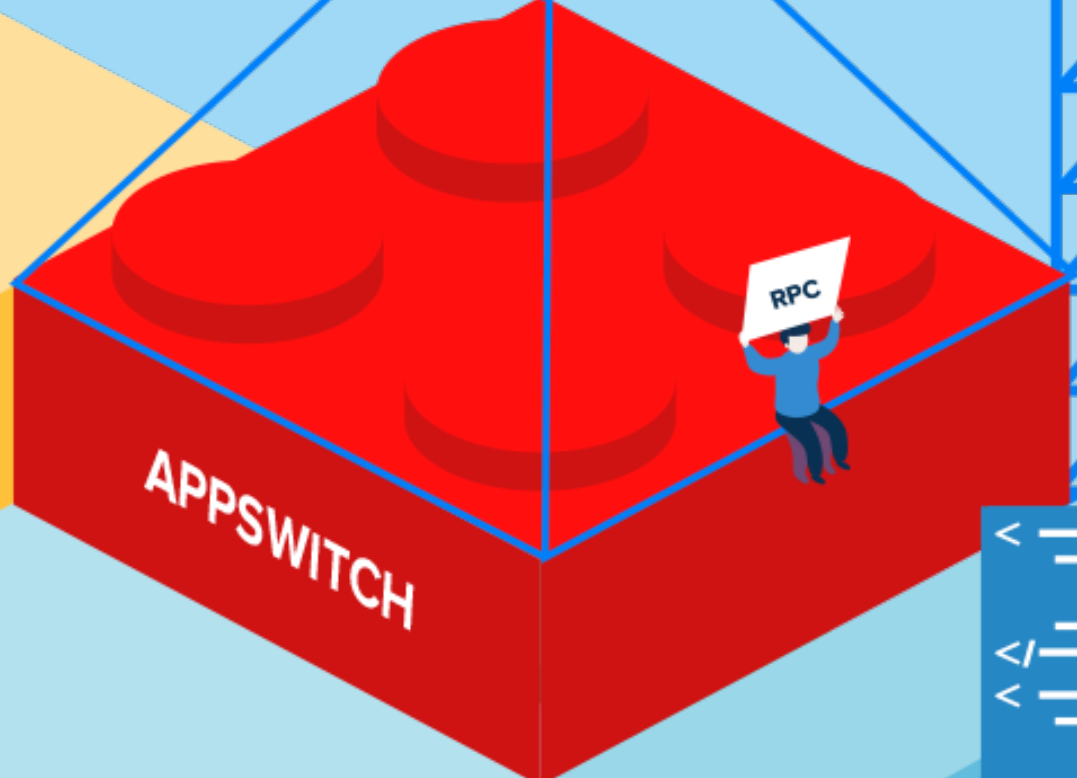
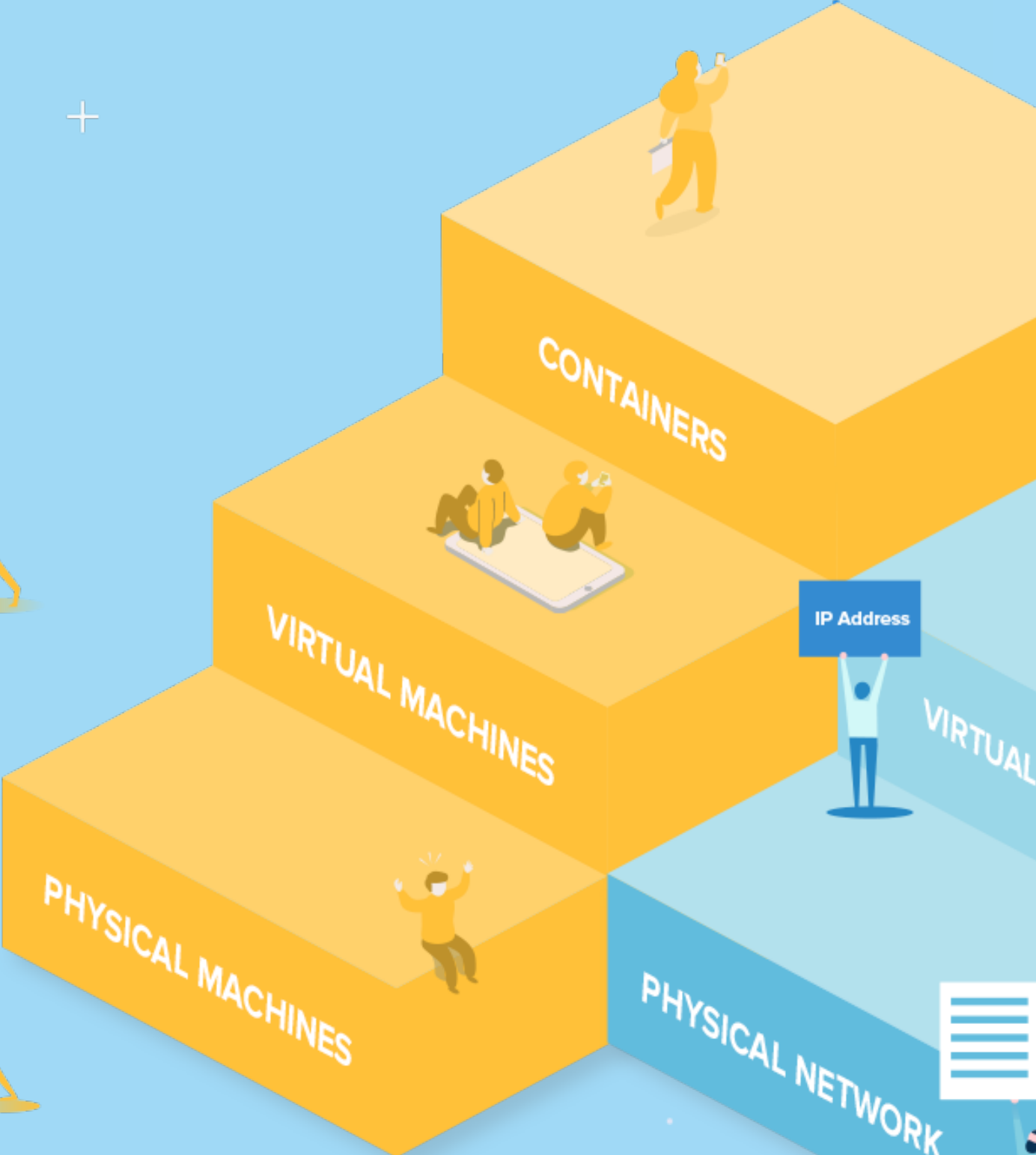


AppSwitch

Application

Virtual

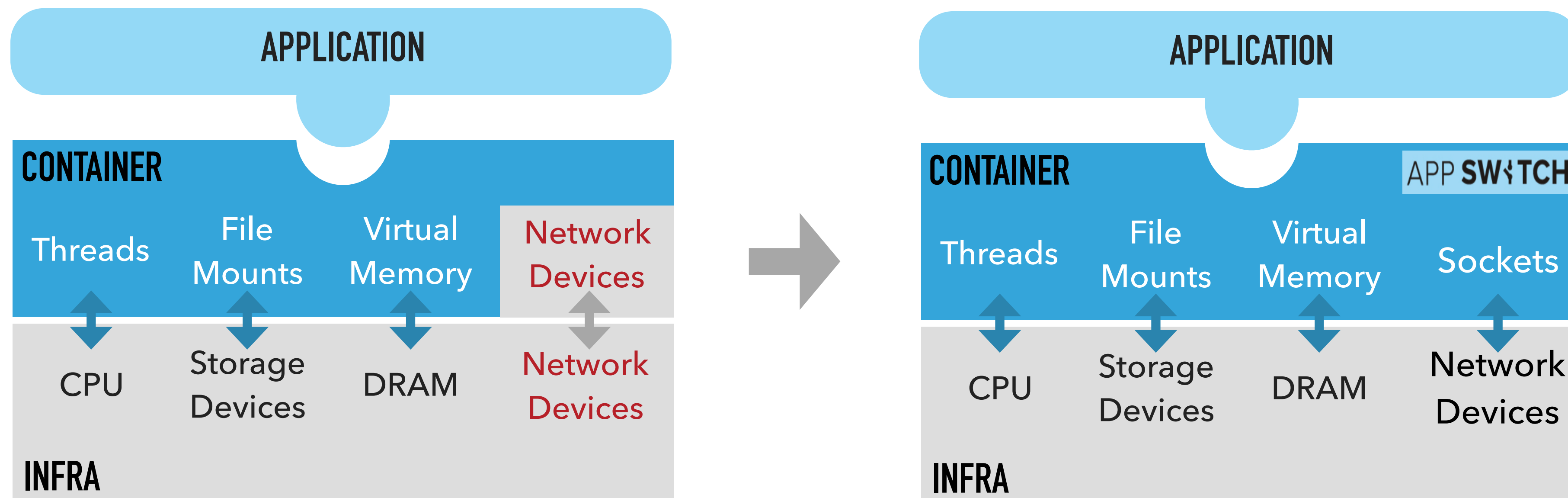
Physical



COMPUTE

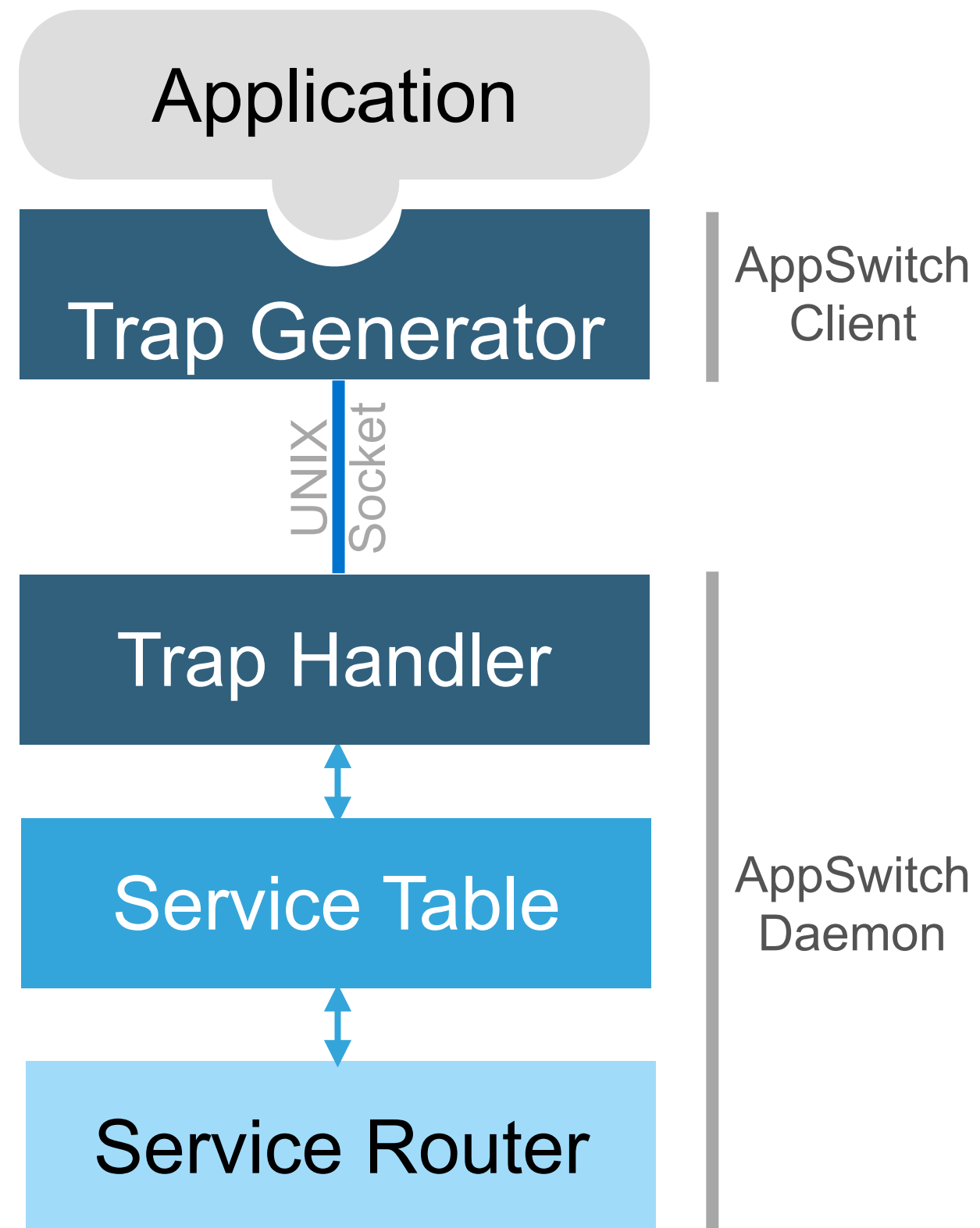
NETWORK

AppSwitch – Socket / Port Namespace, Not Device Namespace



AppSwitch enables the container to work with an OS level abstraction for network and remove requirement for additional privilege

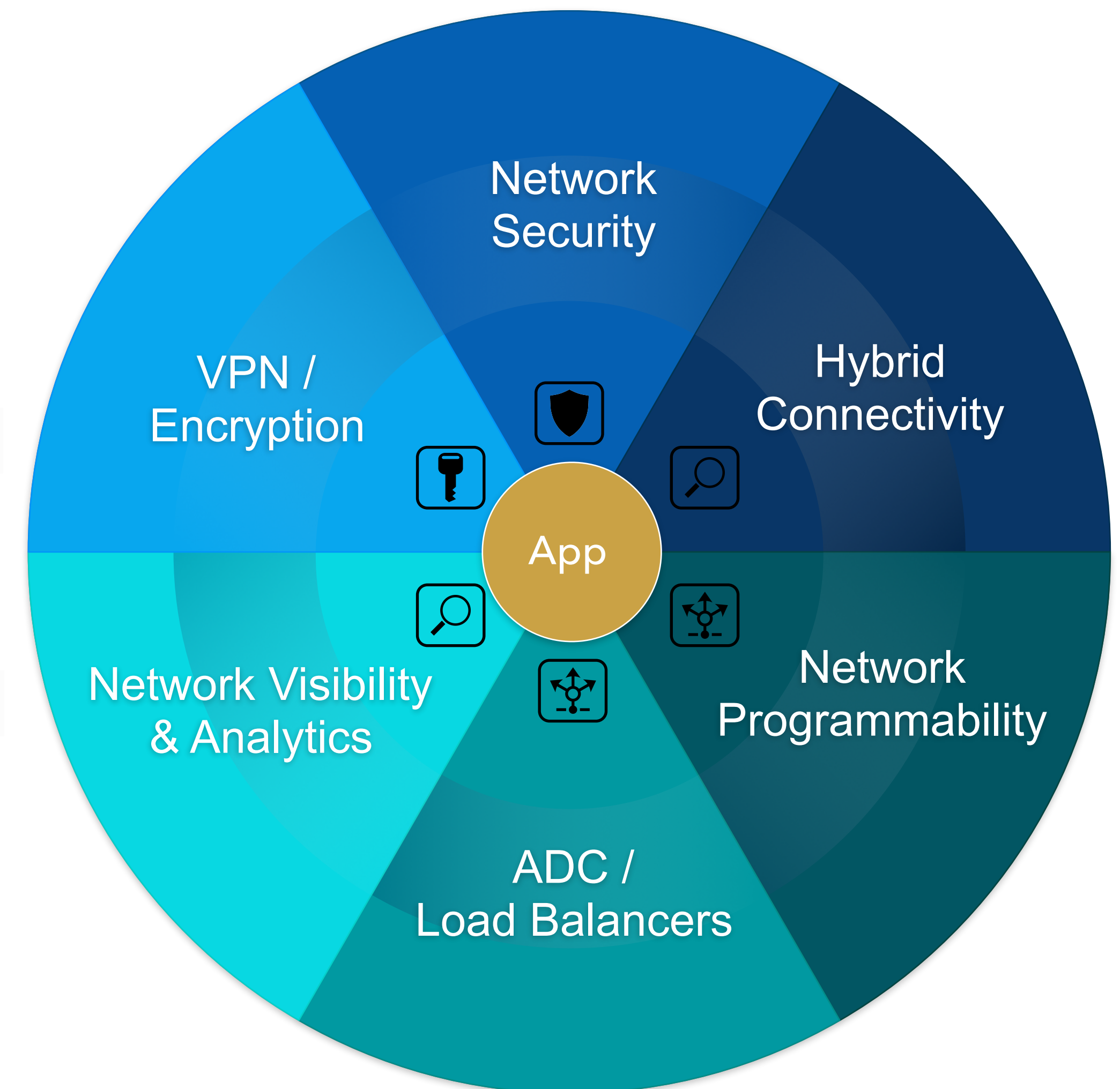
Architecture



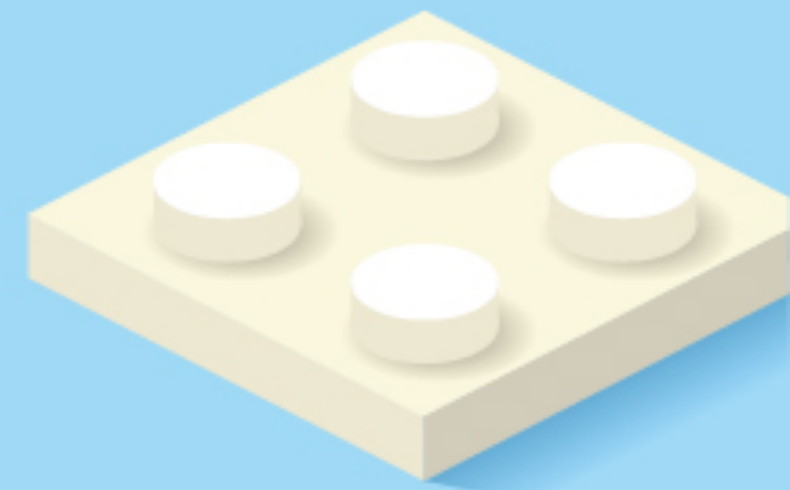
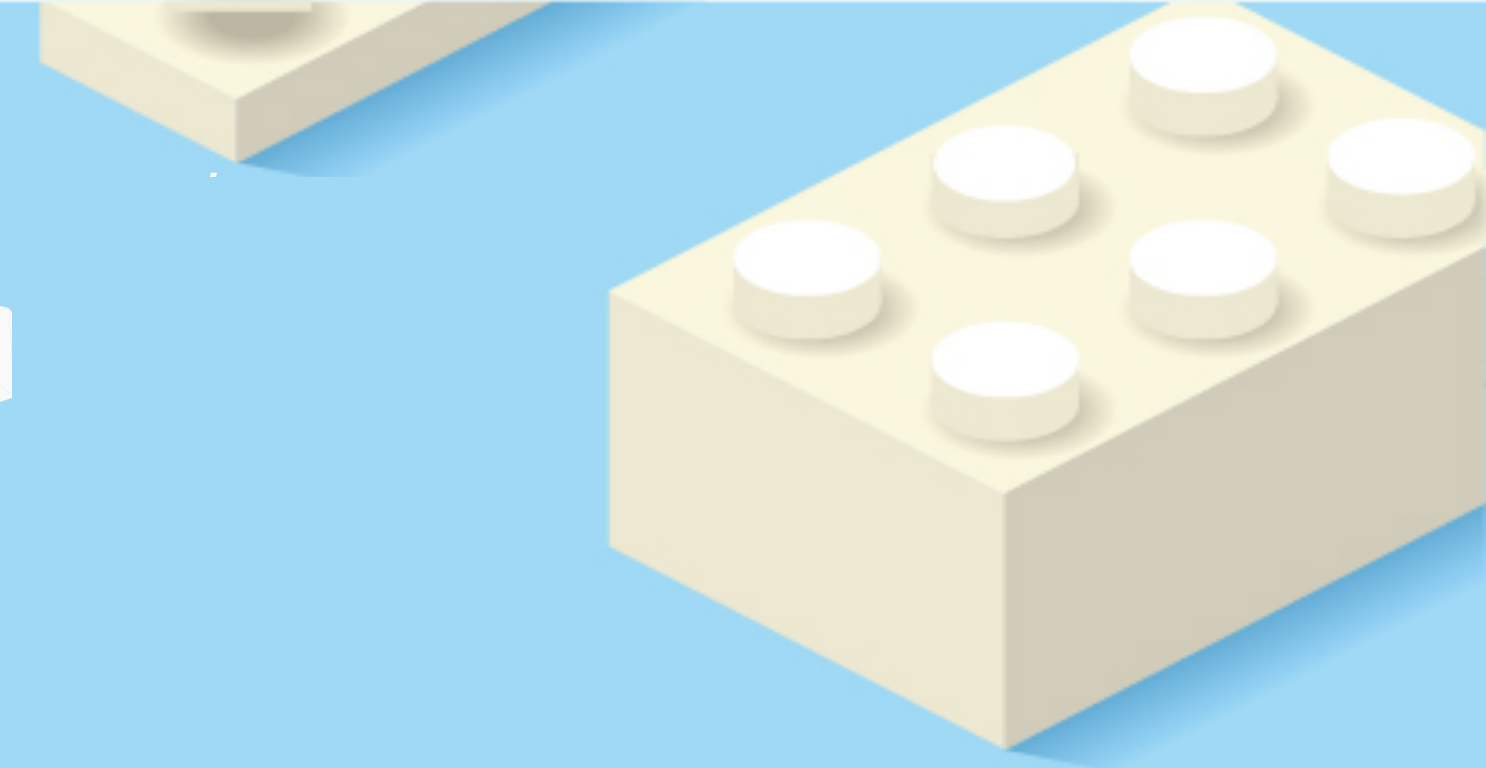
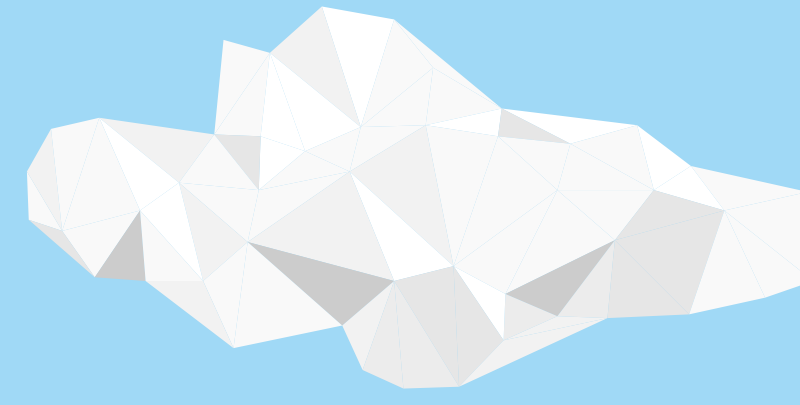
- ▶ One self-contained static binary that serves as client and daemon
- ▶ Client transparently plugs into the application and monitors its network API
- ▶ Daemon handles the API calls forwarded by the client
- ▶ Keeps service table up-to-date
- ▶ Propagates service information across cluster(s)

Some of the other capabilities....

- ▶ Auto-curated service registry with automatic discovery of service dependencies and their ordering
- ▶ Extremely efficient enforcement of label-based access controls without any data path processing.
- ▶ Proxy-less, client-side load balancer
- ▶ IP address portability with ability to assign arbitrary IP addresses to applications regardless of underlying network.
- ▶ Transparently redirect connection requests to alternate services without using NAT in case of application migration.
- ▶ "flat" connectivity with client IP preservation across hybrid network environments.
- ▶ Fine grain observability
- ▶ Transparent TLS insertion
- ▶ Better than native performance



Demo



Try it out

<https://hub.docker.com/r/appswitch/ax/>

References

- ▶ <http://appswitch.io>
- ▶ <https://istio.io/blog/2018/delaying-istio/delaying-istio>
- ▶ <http://jpetazzo.github.io/2018/03/13/appswitch-hyperlay-network-stack-future>
- ▶ <https://networkop.co.uk/post/2018-05-29-appswitch-sdn>
- ▶ http://appswitch.io/blog/kubernetes_istio_and_network_function_devirtualization_with_appswitch
- ▶ <http://hci.stanford.edu/cstr/reports/2017-01.pdf>
- ▶ <http://appswitch.readthedocs.io/en/latest>
- ▶ https://www.youtube.com/watch?v=LPKpR_Fitn8
- ▶ <https://www.youtube.com/watch?v=C55KjvjLYnE>

The screenshot shows a web browser window with the URL `appswitch.io`. The page title is "It's Time to Devirtualize the Network". The main text reads: "AppSwitch - a.k.a. `ax` is a simple tool that automatically infuses next generation networking capabilities directly into your existing applications." Below the text are two buttons: "Download" and "Github".

The diagram below the text illustrates a multi-layered architecture. On the left, a vertical axis is labeled "Application", "Virtual", and "Physical". On the right, a vertical axis is labeled "COMPUTE" and "NETWORK". The diagram shows a stack of layers: "PHYSICAL MACHINES" at the base, followed by "VIRTUAL MACHINES", "CONTAINERS", and "APPSWITCH" (represented by a red block). A "VIRTUAL NETWORK" layer is shown between the "VIRTUAL MACHINES" and "CONTAINERS" layers. A "PHYSICAL NETWORK" layer is shown at the bottom. A crane is lifting a red block labeled "JSON configuration" into the "APPSWITCH" layer. The diagram also includes icons for "IP Address" and "VLAN ID".